

# Towards AI Methods for Clause Selection

Jan Jakubův<sup>1</sup>   Josef Urban<sup>1</sup>   Bob Veroff<sup>2</sup>

<sup>1</sup>Czech Technical University in Prague

<sup>2</sup>Univeristy of New Mexico

AITP17, Obergurgl, 29th March 2017

# Outline

- 1 Introduction
- 2 Proof Sketches
- 3 Enigma Models
- 4 Conclusion

# The Automated Mathematician

Should at least be able to ...

- consistently and reliably prove “easy” problems easily
- formalize proofs of existing mathematical knowledge
- serve as an effective assistant or collaborator for the research mathematician

... and eventually could even propose and prove interesting and useful things.

As we are hearing at this conference, current research efforts focus on different aspects of these ambitious goals and are exploring a variety of methods.

# Given Clause Loop Paradigm

## Problem representation

- first order clauses
- posed for proof by contradiction

Given an initial set  $C$  of clauses and a set of inference rules, find a derivation of the *empty clause* (for example, by the resolution of two conflicting clauses  $P$  and  $\neg P$ ).

## Basic loop

```
while (no proof found)
{
    select a given clauses
    apply inference rules to selected clauses
    process inferred clauses
}
```

A common variation is to postpone some processing of inferred clauses until they are selected (“Otter Loop” vs. “Discount Loop”).

# Clause Selection Methods

## Selection mechanisms

- symbol count (weighting)
- user-defined weighting patterns
- attribute-based restrictions (e.g., set of support)
- model-based selection (semantic guidance)
- subsumption-based selection (e.g., hints)
- selection by learned classifier

User-defined rules or scripts can be used to specify combinations of these mechanisms.

# Two Approaches To Learning Clause Selection

- Proof sketches - a symbolic approach
- ENIGMA - a statistical learning approach

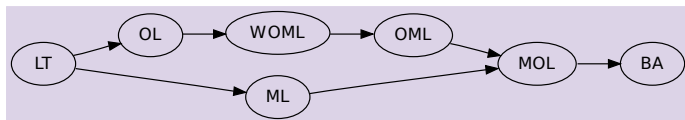
# Outline

- 1 Introduction
- 2 Proof Sketches**
- 3 Enigma Models
- 4 Conclusion



# Theory Hierarchies

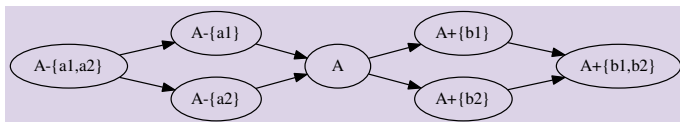
Example: Lattice Theory ( $\mathcal{LT}$ )



$\mathcal{LT}$

- + Invertibility and Compatibility ( $\mathcal{OL}$ )
- + Weak Orthomodularity ( $\mathcal{WOML}$ )
- + Orthomodularity ( $\mathcal{OML}$ )
- + Modularity ( $\mathcal{MOL}$ )
- + Distributivity ( $\mathcal{BA}$ )

# Theory Hierarchies



What extensions to a theory suffice to yield a proof of theorem  $t$ ?

How far up the hierarchy can we push theorem  $t$ ?

How can we use the hierarchy to help find a proof of  $t$  in  $A$ ?

- Use models in weakenings of  $A$ .
- Use proofs in extensions of  $A$ .

# Problem Selection

Eliminating an assumption can lead to new conjectures.

- Adding new assumptions
- Weakening current assumptions (e.g., instances)

$$(x * x) * (y * y) = (y * y) * (x * x)$$

- Unanticipated properties

$$K(K(x, y), z) = K(x, K(y, z))$$

# Hints Selection

Criteria for hint selection and prioritization:

- domain knowledge
- most recent history
- frequency counts
- set closeness measures
- mutual information

## Success and Progress

Publications in respected math journals:

*Algebra Universalis, Journal of Algebra, Transactions of the AMS, Notre Dame Journal of Formal Logic, Studia Logica, ...*

Proof lengths have been increasing over time:

2011:	2015:	2017:
24,356	73,625	242,134
18,862	69,489	141,589
17,075	54,742	112,135
16,400	45,131	89,716
15,785	40,708	87,534

# Outline

- 1 Introduction
- 2 Proof Sketches
- 3 Enigma Models**
- 4 Conclusion


# Enigma Models

**Idea:** Lets use fast linear classifier to guide given clause selection!

**ENIGMA:** Efficient learNing-based Inference Guiding MACHine

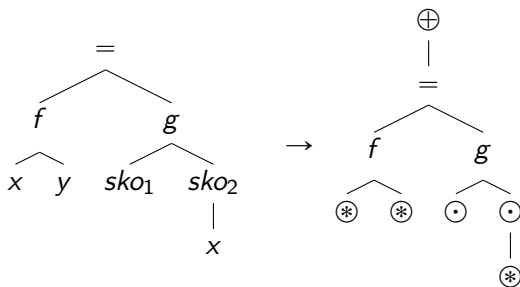
- LIBLINEAR: open source library<sup>1</sup>
- **input:** positive and negative examples (float vectors)
- **output:** model ( $\sim$  a vector of weights)
- **evaluation** of a generic vector: dot product with the model

---

<sup>1</sup><http://www.csie.ntu.edu.tw/~cjlin/liblinear/> 

# Clauses as Feature Vectors

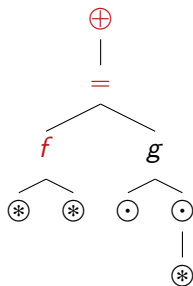
Consider the literal as a tree:





# Clauses as Feature Vectors

Count descending paths of length 3 (our features):



$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

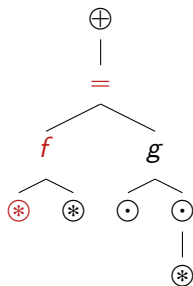
$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

# Clauses as Feature Vectors

Count descending paths of length 3 (our features):



$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

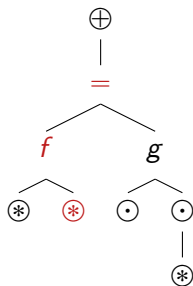
$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

# Clauses as Feature Vectors

Count descending paths of length 3 (our features):



$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

## Clauses as Feature Vectors

Number features and construct a (sparse) vector:  
 (each feature has a fixed position,  $|features| = |\Sigma|^3$ )

$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

$$(0, \dots, 1, \dots, 1, \dots, 2, \dots, \\ 2, \dots, 1, \dots, 0, \dots)$$

# Clauses as Feature Vectors

Number features and construct a (sparse) vector:  
 (each feature has a fixed position,  $|features| = |\Sigma|^3$ )

$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

$$(0, \dots, 1, \dots, 1, \dots, 2, \dots, \\ 2, \dots, 1, \dots, 0, \dots)$$

## Clauses as Feature Vectors

Number features and construct a (sparse) vector:  
 (each feature has a fixed position,  $|features| = |\Sigma|^3$ )

$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, \otimes) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, \otimes) \mapsto 1$$

$$(0, \dots, 1, \dots, 1, \dots, 2, \dots, \\ 2, \dots, 1, \dots, 0, \dots)$$

## Clauses as Feature Vectors

Number features and construct a (sparse) vector:  
 (each feature has a fixed position,  $|features| = |\Sigma|^3$ )

$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

$$(0, \dots, 1, \dots, 1, \dots, 2, \dots, \\ 2, \dots, 1, \dots, 0, \dots)$$

# Clauses as Feature Vectors

Number features and construct a (sparse) vector:  
 (each feature has a fixed position,  $|features| = |\Sigma|^3$ )

$$(\oplus, =, f) \mapsto 1$$

$$(\oplus, =, g) \mapsto 1$$

$$(=, f, *) \mapsto 2$$

$$(=, g, \odot) \mapsto 2$$

$$(g, \odot, *) \mapsto 1$$

$$(0, \dots, 1, \dots, 1, \dots, 2, \dots, \\ 2, \dots, 1, \dots, 0, \dots)$$



# ENIGMA inside E Prover

- E Prover uses clause weight functions
- ... the clause with the smallest weight is selected
- Implemented new weight function Enigma
- ... parametrized by a given model (argument)
- Enigma can be composed with other weight functions:

```
(100*Enigma(modelX, args)  
 5*OtherWeight(args))
```

# Experiments Setting

- AIM problems from CASC (LTB category)
- ... 1020 training problems, 200 testing problems
- advantages (simplifications):
- ... different conjectures in the same theory
- ... small number of symbols (8+4)
- ... symbols are used consistently

# Manual Evaluation

- 1 Use single E strategy  $S$  to solve 239 training problems (in 30s)
- 2 Extract training examples from solved problems:
  - positive: processed clauses used in proofs
  - negative: processed but not used in proofs
- 3 Create Enigma model  $M$
- 4 Combine  $M$  with  $S$  in different ways ( $S_1, \dots, S_{17}$ )  
( $M$  alone or plugged into  $S$ ; different frequencies)


# Manual Evaluation Results

- Original  $S$  (without Enigma) solves 22 problems in 180s
- Best  $S_i$  (with Enigma) solves 41 problems
- Other  $S_i$ 's? ( $3 \times < 22$ ,  $5 \times \geq 40$ )
- All  $S_i$ 's together solves 52 (in  $17 * 180$  s)
- Only 3  $S_i$ 's covers 52 solved problems
- (for comparison: Vampire solves 47 in  $3 * 180$ s)

# Automated Strategy Development

- BliStrTune<sup>2</sup>: System for developing E strategies
  - provide set of benchmarks
  - run it for few days (7) on few cores (32)
  - collect new E Prover strategies
- EnigmaTuner = BliStrTune + Enigma
  - develop Enigma models on the way
  - incorporate Enigma models into other strategies

---

<sup>2</sup><http://github.com/ai4reason/BliStrTune> 

## Preliminary Results (work in progress)

- Select protocols best-performing on training problems
- Run them in sequence giving the time limit
- Compare with E, Vampire, Prover9 in 300 seconds

prover	solved	SOTA+
E 1.9.1 (EnigmaTuner)	77	+37%
Prover9 (no hints)	56	+0%
Vampire 4.0 (CASC)	45	-19%
E 1.9.1 (auto-schedule)	31	-44%

# Outline

- 1 Introduction
- 2 Proof Sketches
- 3 Enigma Models
- 4 Conclusion**

# Future Work and Challenges

- Better ways to incorporate BliStrTune with Enigma
- Generalize to different theories
- Need training on “similar” problems
- Deal with large theory signatures
- Deal with different term orderings, etc.
- Deal with inconsistent symbols (e.g. TPTP)



# Thank you

Questions?